ÉCOLE SUPÉRIEURE D'INGÉNIEURS DE LUMINY
UNIVERSITÉ DE LA MÉDITERRANÉE AIX-MARSEILLE II

Cahier des charges
Projet ManateeX

Jeu de Tower Defense

Thomas Fourcroy & Benoit Djerigian 28/11/11

Sommaire

Introduction

1. Description du jeu

Le jeu de Tower Defense (TD) consiste à défendre un objectif contre des vagues d'ennemis. Pour cela, le joueur peut construire des éléments défensifs qui peuvent gêner la progression des ennemis et/ou les détruire. Les ennemis se dirigent aussi vite que possible vers l'objectif et l'attaquent dès qu'ils sont à portée. Le jeu se termine si le joueur élimine toutes les vagues d'ennemis sans perdre l'objectif (victoire) ou si l'objectif est détruit (défaite).

Les TD sont nés avec les premiers jeux de stratégie proposant un éditeur de cartes, tel Age of Empires. Les bases étaient déjà posées : possibilité de construire des bâtiments, dont des tourelles défensives, présence de nombreuses unités de force variable pouvant attaquer les bâtiments, etc. Il suffisait donc de scripter le comportement des ennemis contrôlés par l'IA, les conditions de victoire et de défaite, et de modifier le cas échéant les caractéristiques des éléments du jeu.

Deux jeu de stratégie développés par Blizzard Entertainment, Starcraft puis Warcraft III, ont grandement contribué à l'essor du TD. D'une part grâce à leurs éditeurs de cartes puissants, d'autre part grâce à leur succès auprès des joueurs.

Le Flash a également permis la démocratisation du TD sur navigateur. On en trouve également sous forme de jeux complets, payants ou gratuits, sur PC ou smartphone.

Il existe deux variantes au TD : avec ou sans mazing. Ce terme désigne la possibilité laissée au joueur de ralentir les ennemis en construisant un labyrinthe.

- Dans un TD sans mazing, le chemin emprunté par les vagues d'ennemis n'est pas constructible. Le joueur place des tours offensives en bordure du chemin et tente de détruire les ennemis avant qu'ils n'atteignent l'objectif.
- Dans un TD avec mazing, le joueur construit directement sur le chemin emprunté par les ennemis. Il est donc libre de disposer ses éléments de manière à rallonger le chemin vers l'objectif. Dans ce cas de figure, il lui est interdit de bloquer totalement l'accès à l'objectif.

Le projet ManateeX consiste à concevoir un jeu de Tower Defense avec mazing.

2. Compétences requises

- Conception d'algorithmes
- Représentation UML
- Programmation en langage C++
- Connaissances dans le domaine des jeux.

3. Motivations

Nous avons fait le choix de développer un TD car c'est un type de jeu que nous aimons tous les deux. Il correspond aux exigences du projet de programmation objet car il consiste à programmer une IA gérant les interactions entre ennemis, défenses et objectif, avec une phase de jeu entre chaque vague durant laquelle le joueur est autorisé à mettre en place les éléments défensifs pour la prochaine vague. Le rôle du joueur du point de vue programmation est donc purement un rôle de création d'objets, qui interagiront par la suite de manière automatique.

Terminologie

Joueur : Utilisateur du logiciel, son rôle est de construire des éléments défensifs.

Objectif : Bâtiment que le joueur doit défendre.

PV : Points de Vie, la quantité de dégâts qu'un objet peut encaisser avant d'être détruit.

Ennemi : Objet du jeu mobile qui cherche à atteindre et détruire l'objectif. Il existe plusieurs types d'ennemis en fonctions de leurs caractéristiques propres (vitesse de déplacement, quantité de PV, force,...)

Tour : Élément défensif capable d'attaquer les ennemis. Peuvent-êtres de plusieurs types différents en fonctions de leurs caractéristiques propres (cadence de tir, puissance...)

Niveau : Indique le niveau d'évolution des objets. Le passage d'un niveau améliore une ou plusieurs caractéristiques d'un objet.

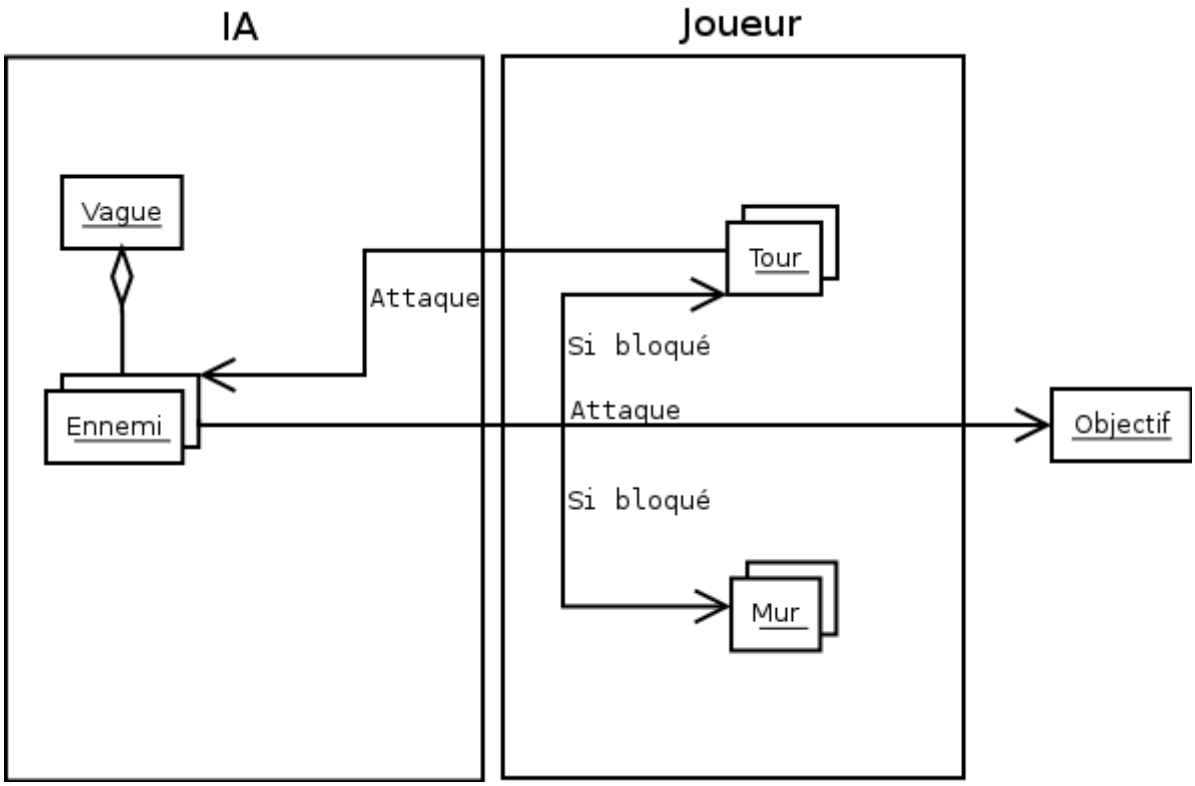
Plateau : indique l'espace visuel où se déroule le jeu. Peut contenir certains éléments de décor.

Mur : élément défensif passif construit par le joueur. Il sert a forcer les ennemis à emprunter un certain chemin.

Modes : il existe deux modes de jeu :

-normal : le joueur gagne après avoir repoussé un nombre donné de vagues,
-survivor : des vagues d'ennemis continuent d'apparaître jusqu'à ce que l'objectif soit détruit.

Architecture générale



Le joueur construit des éléments défensifs (tours ou murs) afin de défendre l'objectif. À intervalle régulier, l'IA fait apparaître une vague d'ennemis qui se dirige vers l'objectif afin de l'attaquer. Les murs et les tours forcent les ennemis à dévier de leur trajectoire, les tours les attaquent quand ils sont à portée.

Dans le cas où il n'existe pas de chemin pour atteindre l'objectif car l'accès est bloqué par des éléments défensifs, les ennemis peuvent alors attaquer le mur où la tour la plus proche pour ouvrir un nouveau chemin.

Architecture détaillée

1. Fonctionnalités

Le jeux devra avoir les fonctionnalités suivantes :

- Écran d'accueil permettant le lancement du jeu ainsi que l'accès aux autres fonctionnalités (options, scores, modes...)
- Écran d'options permettant de régler les différentes options du jeux, comme la difficulté
- Écran de modes permettant de choisir entre le mode normal et survivor
- Écran des scores affichant le top 10
- Écran de jeux: Représente le "plateau" où se déroule la partie. Le plateau est divisé en NxM cases (à définir) ayant le comportement suivant:
  - Un click sur une case vide ouvre un menu permettant de choisir l'objet défensif à construire
  - Un click sur une case où se trouve un élément défensif ouvre un menu indiquant les informations relatives à ses caractéristique ainsi qu'une paire de bouton: l'un permettant de "vendre" l'élément, l'autre de faire une amélioration d'une ou plusieurs de ses caractéristiques d'un pourcentage donné (et indiqué)
  - Un click sur une case où se trouve un ennemi indique brièvement ses caractéristiques
  - Un click sur l'objectif indique brièvement son état
- Écran de fin de partie indiquant si elle s'est soldée par une victoire ou une défaite, et un ensemble de statistiques (score, ennemis tués, temps, etc...)

Sur les contours du plateau sont disposés divers boutons (pause, avance rapide, menu du jeu,...) ainsi que des indications sur la partie en cours (score, PV de l'objectif, nombre de vagues restantes, temps,...)

2. Contraintes techniques

Le projet de jeu vidéo défini dans ce cahier des charges devra être écrit en langage C++.

Le logiciel devra fonctionner sous Linux (au minimum, portage vers d'autres plate-formes à envisager).

Une interface graphique minimale est également requise, cependant le choix de la librairie à utiliser reste libre. Notons également que, quelque soit le choix des graphismes, la performance générale doit être le fil conducteur de ce projet. Le jeu devra être fluide (pas de saccades) et répondre aux interactions en un temps imperceptible pour le joueur.

Livrables

Les développeurs auront à leur charge les fournitures suivantes:

- Le(s)logiciel(s) objet(s) du présent cahier des charges, fonctionnant sur les plate-formes précédemment citées
- Un manuel d'utilisation, avec règles du jeu
- Un rapport détaillé de développement
- Tous les fichiers sources dûment documentés

La documentation sera fournie en version papier et numérique. Les codes source seront fournis au format numérique.

Délais

Les applications et documentations associées devront être réceptionnées au plus tard le 31 Janvier 2012.

Licence - Propriété

Ce logiciel une fois réalisé sera disponible exclusivement sous licence GNU/GPLv3 dont la version juridique complète est disponible à cette adresse :

http://www.gnu.org/licenses/gpl.html

ou une traduction en français:

http://org.rodage.com/gpl-3.0.fr.html